# Extendable Bits of Digital On/Off Signal Controlling Using Multi-Platform Clients via Web Services to Single Arduino Output Port by Using SIPO Shift Registers

Kayun Chantarasathaporn[1], Sudasawan Ngammongkolwong[2], Songpol Nakarateruangsuk[3] and Chom Kimpan[4]

[1] Faculty of Business Administration
Huachiew Chalermprakiet University
Samutprakarn, Thailand
dr.kayun@gmail.com
[2, 3] Faculty of Science and Technology
Southeast Bangkok College
Bangkok, Thailand
sudasawan@southeast.ac.th, songpol@southeast.ac.th
[4] Faculty of Engineering and Technology
Panyapiwat Institute of Management
Nonthaburi, Thailand
kkchom@kmitl.ac.th

**บทคัดย่อ** --- ช่วงหลายทศวรรษที่ผ่านมา สัญญาณดิจิตอลได้ถูกนำเข้ามาใช้ในการเฝ้าดูและควบคุมในอุตสาหกรรมหลากหลายระบบ ในอดีต กระบวนการจัดการเรื่องนี้ถือได้ว่าเป็นสิ่งซับซ้อน แต่หลังจากที่มีการเกิดขึ้นของไมโครคอนโทรลเลอร์งานเหล่านี้จัดได้ว่าง่ายขึ้น และยิ่งง่ายในการศึกษาความรู้แขนงนี้ขึ้นไปอีกเมื่อมีการเกิดขึ้นของอาดูอิโนซึ่งเป็นบอร์ดไมโครคอนโทรลเลอร์สำเร็จรูปที่มีสถาปัตยกรรมแบบเปิด

กระนั้นอาดูอิโนก็มีข้อจำกัดที่จำนวนพอร์ตดิจิตอลสำหรับนำสัญญาณเข้าและออก โครงงานนี้ได้พยายามแก้ไขข้อจำกัดนี้โดยการประยุกต์เทคโนโลยี SIPO ชิฟรีจิสเตอร์เข้ากับอาดูอิโน เนื่องจากนักพัฒนาสามารถนำชิฟรีจิสเตอร์มาเชื่อมกันได้แบบต่อเนื่อง ในทางทฤษฎีแล้วจำนวนบิตของสัญญาณดิจิตอลขาออกจึงไม่ถูกจำกัด (ขยายได้ไม่จำกัด)

ในมุมมองของการพัฒนาซอฟต์แวร์ โครงงานนี้ใช้สถาปัตยกรรมซอฟต์แวร์แบบ ๒ ชั้น โดยในส่วนตรรกะธุรกิจซึ่งใช้เป็นซอฟแวร์คั่นกลางได้เลือกใช้เทคโนโลยีเว็บเซอร์วิสเนื่องจากเป็นมาตรฐานเปิดที่สามารถสนับสนุนซอฟต์แวร์ฝั่งผู้ใช้จากเทคโนโลยีและแพลตฟอร์มที่หลากหลาย เว็บเซอร์วิสเซิร์ฟเวอร์ประกอบด้วยเว็บเมธอดสำหรับแลกเปลี่ยนข้อมูลกับโปรแกรมของอาดูอิโน

ซอฟต์แวร์ฝั่งผู้ใช้สำหรับควบคุมอุปกรณ์ผ่านเว็บเซอร์วิสในโครงงานนี้ยกตัวอย่างไว้ ๒ แพลตฟอร์มคือซอฟต์แวร์แบบวินโดวส์ฟอร์มและ ASP.NET เว็บฟอร์ม ซึ่งการนี้ทำให้ผู้ใช้สามารถทำงานกับระบบได้ทั้งจากเครือข่ายภายในและเครือข่ายระยะไกล

***คำสำคัญ: สัญญาณดิจิตอล, ควบคุม, ชิฟรีจิสเตอร์, ไมโครคอนโทรเลอร์, อาดูอิโน, เว็บเซอร์วิส, วินโดวส์, เว็บ***

*Abstract* --- Several decades ago, digital signal has been embraced to the industry for monitoring and controlling various systems. In the past, to handle these jobs, the processes were rather complicated. However, after having microcontroller, these tasks were easier. It has been even much easier to learn this issue after the launch of Arduino which is an instance open microcontroller board.

However, Arduino has some limitation in the number of digital input and output port. This project tries to overcome this constraint by applying SIPO Shift Register technology with Arduino. As developers can cascade Shift Registers, theoretically, the numbers of bit of digital output are not limited.

From the software viewpoint, this project is 2-tier application. For business logic that works as a middleware, Web Services is chosen because it is open standard that can serve clients from various technologies and platforms. Web Services server contains Web Methods for exchanging data with Arduino program.

Client applications for controlling devices through the Web Services are provided in 2 platforms, Classic Windows Form Application and ASP.NET Web Form Application. It means users can work from both local and remote networks.

***Keywords - digital signal; control; shift register; microcontroller; arduino; web services; windows application; web application***

## I. INTRODUCTION

Sending control signal is one of the famous task in the industry. This study is going to let multi-platform clients be able to send control signals which are multi-bit Digital On/Off data via just one output port at microcontroller. Using multi-platform client in this case can be done conveniently because the project is designed to separate between business logic and user interface.

The contents of this paper are as follows. First is introduction. Second is overall picture. Third is brief explanation about related knowledge used in the project. Fourth is section of development processes and outputs. Fifth is conclusion. The last, sixth, mentions about future works.

To explain easier about this project, the author would like to divide the whole system to 4 parts. First is electronic device part that is considered pure hardware. Second is microcontroller part which contains both software and hardware. Third is Web Services part that works as business logic software. Final part, fourth, is the client software which works as user interface. Client can be any kinds of platform from various technologies that support Web Services, but this project chooses Windows Form and Web Form Applications.

## II. PROJECT OVERVIEW

### A. Client Software

This project separates graphic user interface and business logic apart from each other. The business logic is located at Web Services. Sample client applications in this case are in 2 platforms, Classic Windows Form Application and ASP.NET Web Form Application. Client gets data that are intended to send to control devices from user via user interface screen.

### B. Web Services for Sending Control Signals

Sending Digital On/Off control signals to the target devices was extended from watching at the terminal in front of the system to computer network. One of the standard methodology for remote controlling and monitoring over TCP/IP network is Web Services [1]. Remote procedure calls via Web Services has been well accepted because they can use standard Web port (80 or 443) as the carrier. These two ports are usually transparent for all firewall. The W3C defines a Web service generally as “a software system designed to support interoperable machine-to-machine interaction over a network”.

Web Services, in this project, gets data from Arduino and runs on Microsoft Internet Information Server (IIS) Web Server.

### C. Microcontroller

This project chooses Arduino[2] as the brain of the system because it is low price with acceptable performance and open source in both software and hardware. Arduino provides both Software Development Kit (SDK) and Integrated Development Environment (IDE). By the way, many serious professional developers may prefer popular IDE such as Microsoft Visual Studio with Visual Micro's Arduino plug-in instead since it has much more helping tools, such as, Intellisense (code completion) and debugging.

### D. Electronic Devices

One port per one bit is the easiest way for sending Digital On/Off signal controlling. However, it is hardly possible because the microcontroller board has limited number of port. To overcome this limitation, there are several techniques, such as, using multiplexers[3] or Shift Registers[4]. This study chooses the Shift Registers because they are simple and can support asynchronous connection. Another reason is Shift Registers can be cascaded which makes it easy to enhance the solution in case that the users want to increase number of data bits.

## III. RELATED KNOWLEDGE

To implement this project, there are at least 5 fields of knowledge needed to know, Classic Windows Form Application, ASP.NET Web Form Application, .NET Web Services, Arduino instance microcontroller board and SIPO Shift Register[5]. Brief explanation of them will be demonstrated as follows.

### A. Classic Windows Form Application and ASP.NET Web Form Application

Windows Form application has been popular for a long time before an emerging of the Internet. It can run standalone. There are many languages used for creating Windows Form application, but this project uses C#

ASP.NET Web Form is one of the techniques for creating Web Application provided by Microsoft. Others are like Razor, MVC, however, this project chooses Web Form because it is most concise.

ASP.NET Web Form application usually contains 3 kinds of major file extensions. ".aspx" is for graphic user interface, ".aspx.cs" is for C# code and ".config" is for storing configurations.

### B. .NET Web Services

The W3C Web Services Architecture Working Group defined a Web Services Architecture that it required working with at least the following components

- An interface described in a machine-processable format which was usually called as Web Services Description Language (WSDL).
- The way defined for other systems to be able to interact with the Web Services. How to do is a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages.

Usually, Web Services work through the network by using HTTP protocol with XML serialization along with other Web-related standards. Brief working steps in Web Services Architecture from Figure 1 are as follows

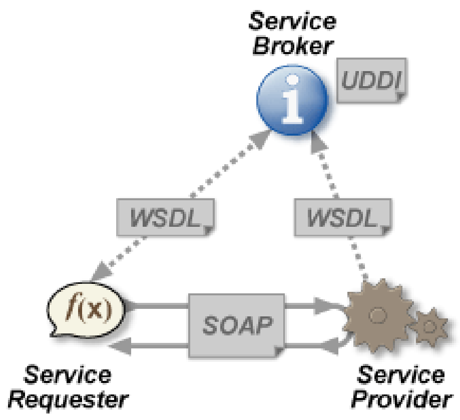


*Figure 1 Web Services Components (https://upload.wikimedia.org/wikipedia/commons/4/4a/ Webservices.png)*

- Service Provider registers its services with Service Broker UDDI and let UDDI advertise this information.
- Service Requester starts searching required service by contacting UDDI to look for appropriated services. UDDI provides WSDL of the appropriated service to the Service Requester.
- After Service Requester gets appropriated WSDL, it can contact Service Provider directly.
- From then, Service Requester and Service Provider can exchange their data by using SOAP (Simple Object Access Protocol) as a wrapper. SOAP is like an envelope that stores real data inside. Data used in Web Services Architecture are often in the format of XML (eXtensible Markup Language).

.NET Web Services is Web Services framework provided by Microsoft. It follows Web Services standard requirements. However, with tools provided in Visual Studio, creating Web Services is quite convenient. .NET Web Services is created based on ASP.NET.

### *C. Arduino Microcontroller Board and SDK*

Arduino is a microcontroller board designed with standard I/O layout. One of the most popular Arduino board is Arduino UNO (Figure 2). The prominent benefit of setting standard layout for I/O pins is third party firms can design various extension modules, called as "Shield". Shield can be stacked, too. Most of the shields use electrical power from main Arduino board. So, to implement with shield, developers have to concern about the total required power whether it is enough or not.

Though Arduino SDK has already provided built-in IDE, it is just a basic. Serious developers usually use more advanced IDE with supplement features, such as, IntelliSense (code completion), debugger and version control. This project uses original Arduino SDK, free version of Arduino development plug-in from Visual Micro and Microsoft Visual Studio 2013 Community Edition.

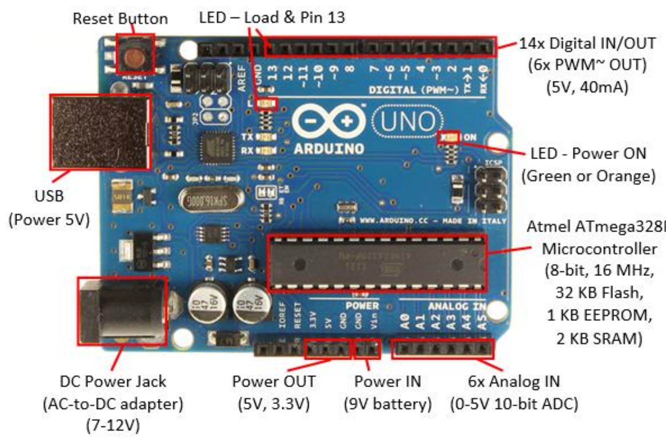
### *D. SIPO Shift Registers*

Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are created by set of flip-flops. Output signal of one flip-flop will be input signal of another flip-flop. Most of the Shift Registers do not have characteristic internal sequence of states. Clock is used for controlling the whole set of flip-flops. All flip-flops are set and reset simultaneously.

There are 4 kinds of Shift Register. Serial In-Serial Out Shift Registers (SISO), Serial In-Parallel Out Shift Registers (SIPO), Parallel In-Serial Out Shift Registers (PISO) and Parallel In-Parallel Out Shift Registers (PIPO)

Serial In Parallel Out Shift Register can be used to get serial data input and distribute them as parallel data output. Sample circuit of SIPO Shift Register is shown in Figure 3. SIPO Shift Register gets serial input data from pin D and feeds data out at pin Q of FF0, FF1, FF2 and FF3 flip-flops respectively. Data from pin Q of FF0 to FF3 are shown as Q0 to Q3. Reading in and feeding out are controlled by clock (CLK) which connects to clock pin on FF0 to FF3. Data from Q0 to Q3 will be fed out at the same time which is considered as parallel data out.

This project uses SIPO to get serial control data as the input and feed out as parallel data. The parallel Digital On/Off control data will be sent to the target devices.



Reset Button
LED – Load & Pin 13
14x Digital IN/OUT (6x PWM~ OUT) (5V, 40mA)
USB (Power 5V)
LED - Power ON (Green or Orange)
Atmel ATmega328P Microcontroller (8-bit, 16 MHz, 32 KB Flash, 1 KB EEPROM, 2 KB SRAM)
DC Power Jack (AC-to-DC adapter) (7-12V)
Power OUT (5V, 3.3V)
Power IN (9V battery)
6x Analog IN (0-5V 10-bit ADC)

Figure 2 Arduino UNO that uses ATmega328 as a CPU (https://www.ntu.edu.sg/home/ehchua/programming/arduino/images/ArduinoUno.png)

## IV. DEVELOPMENT AND OUTPUT

Starting point of this system is at the client, either Classic Windows Form or ASP.NET Web Form Applications as shown in Figure 4. Client, GUI part, sends Digital On/Off control data to .NET Web Services (Web Methods) that locates in IIS Web Server on PC. .NET Web Services, which works as business logic part, feeds the data to Arduino instance microcontroller board. Arduino connects to the PC via USB port. Arduino then sends serial data to CD4094BE SIPO Shift Registers. These Shift Registers convert the serial data to parallel one and send to destination devices. In this project, the destination devices are LED's

TABLE 1 PIN MAP BETWEEN IC AND ARDUINO BOARD

| **Pin# on Most Significant IC** | **Pin# on Least Significant IC** | **Pin# on Arduino Board** |
|---|---|---|
| 1 | 1 | 10 |
| 2 | 9 | |
| 3 | 3 | 11 |
| | 2 | 12 |

### *A. Hardware Portion*

This prototype circuit uses 16 LED's as destination devices for getting Digital On/Off control signals as shown in Figure 5. Data are sent from Web Services on PC to Arduino via USB port at position 1 in the figure. Position 2 is Pin 10 of Arduino works as strobe for controlling CD4094BE Shift Register about what to do between reading data in or writing data out. Position 3 is Arduino's Pin 12 that work as data tube between Arduino and Shift Register. Two CD4094BE SIPO Shift Registers that connect to Digital On/Off destination devices (in this case, they are LED's) are shown at Position 4 in Figure 5 while 16 bit LED's are at Position 5.

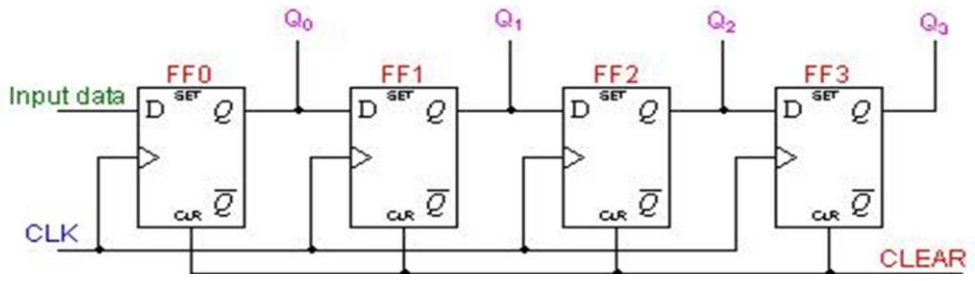


Figure 3 4-bit SIPO Shift Registers.
(http://www.ee.usyd.edu.au/tutorials/digital_tutorial/part2/pics/regist03.jpg)

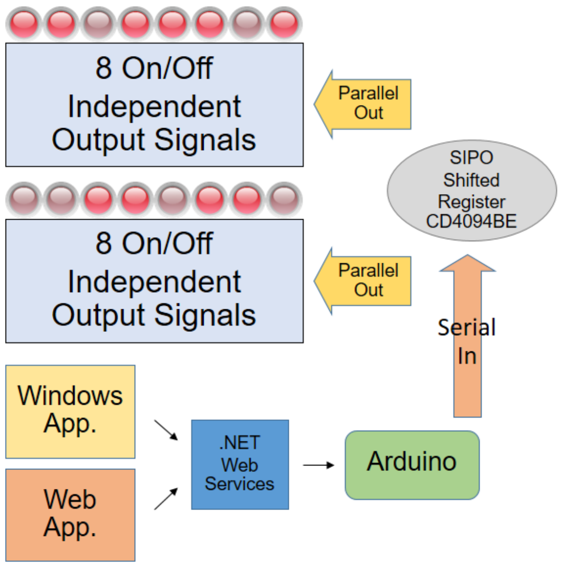

8 On/Off Independent Output Signals
Parallel Out
SIPO Shifted Register CD4094BE
8 On/Off Independent Output Signals
Parallel Out
Serial In
Windows App.
Web App.
.NET Web Services
Arduino

Figure 4 Overview of Controlling System

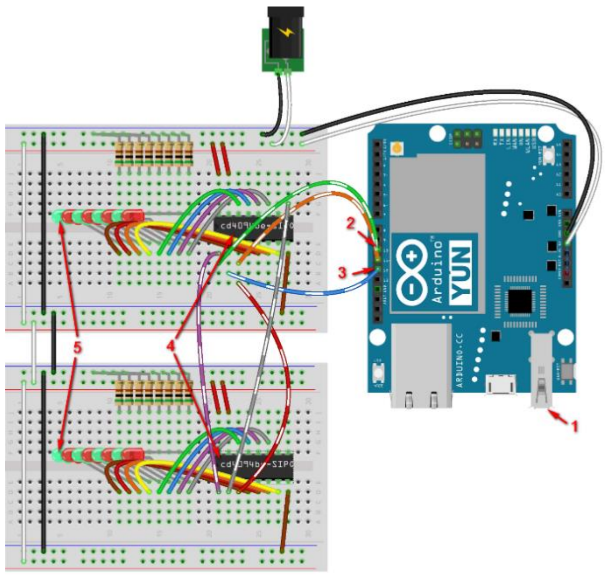
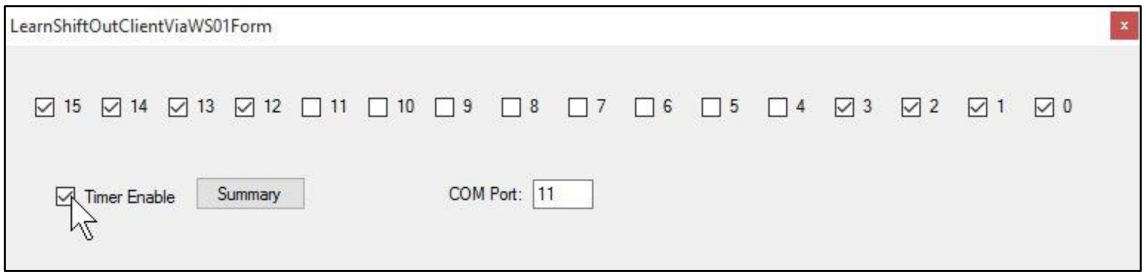


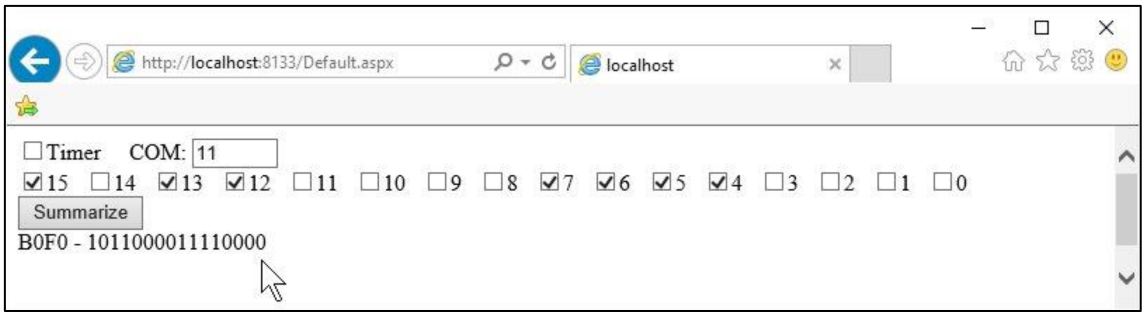
Figure 5 Sending output control signals (Shift Out) by CD4094BE from Arduino


LearnShiftOutClientViaWS01Form

☑ 15 ☑ 14 ☑ 13 ☑ 12 ☐ 11 ☐ 10 ☐ 9 ☐ 8 ☐ 7 ☐ 6 ☐ 5 ☐ 4 ☑ 3 ☑ 2 ☑ 1 ☑ 0

☑ Timer Enable Summary COM Port: 11

Figure 6 Check "Timer Enable" to send summarized data automatically every second.


http://localhost:8133/Default.aspx
localhost

☐ Timer COM: 11
☑ 15 ☐ 14 ☑ 13 ☑ 12 ☐ 11 ☐ 10 ☐ 9 ☐ 8 ☑ 7 ☑ 6 ☑ 5 ☑ 4 ☐ 3 ☐ 2 ☐ 1 ☐ 0
Summarize
B0F0 - 1011000011110000

Figure 7 ASP.NET Web Form Client for sending 16 bit Digital On/Off control data

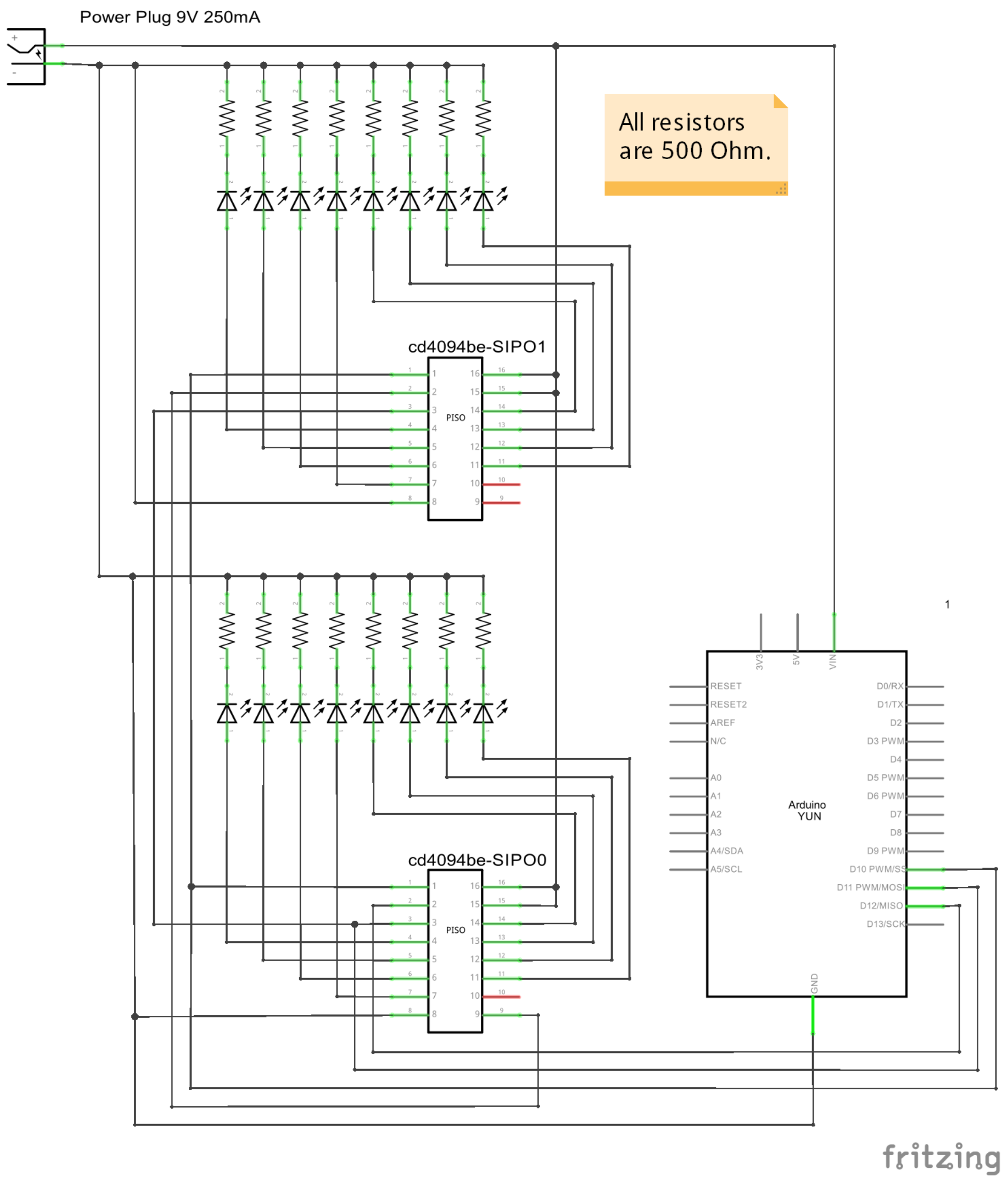


Figure 8 Schematic diagram of the project

In Figure 5, as there are 2 SIPO Shift Registers, one must be assigned as the most significant IC and another will be the least significant one. The above breadboard is the least significant IC while the bottom breadboard is the most significant IC. Both IC's have to be wired together and link to Arduino board as stated pins in Table 1. Schematic diagram of the project is shown in Figure 8.

*B. Software Portion*

There are 3 parts in software portions. (Source code of all software portions can be downloaded from http://goo.gl/gsTPZr)

1) Client Software

This project contains 2 sample kinds of client, Classic Windows Form Application and ASP.NET Web Form Application. Both of them refer to Web Methods in .NET Web Services in 2).

*a) Classic Windows Form Application*

User Interface of this kind of client is as Figure 6. Classic Windows Form Application can run as standalone. Steps of working inside this client are as follows.

(a) Get COM port number that is used for serial communication between PC and Arduino (check from "Device Manager" in "Control Panel" of Microsoft Windows).
(b) Start the program. The screen will be like Figure 6. Fill COM port number that is gotten from (a). Each of sixteen CheckBoxes represent the corresponding 16 bit on/off digital control signals.
(c) After checking the CheckBoxes, manually click the "Summary" button. The collection of on/off data from CheckBoxes will be summarized to be binary string and sent to "SIPOService" Web Services that locates in IIS Web Server.
(d) "SerialPortOpen" Web Method will be called. The parameter for this Web Method is port number. It will open the specified serial port (USB) to be ready for using.

(e) The next Web Method that is called is "SummarizeDigitsAndWrite2Serial". This Web Method needs binary string data, gotten from client, as a parameter. This Web Method will write data to serial port and show Message Box stating the data written in both hexadecimal and binary format. This data will be used by Arduino.

(f) After finish sending, "SerialPortClose" Web Method will be called.

(g) Steps from (b) to (f) are manual working. There is another way that will submit summarized data repeatedly in a period of time. User just check the "Timer Enable" CheckBox as shown in Figure 4.4. If doing like this, the step from (b) to (d) and (f) will be done every second. (Step (e) will be skipped as there will be too many MessageBox).

*b) ASP.NET Web Form Application*

Working logic of the Web Services client in the format of ASP.NET Web Form Application are rather similar to the Classic Windows Form Web Services client. By the way, there is some difference in background technique behind “Timer” control in both platforms. “Timer” in ASP.NET Web Form is not similar to other web controls like what is in Windows Form. Developer needs to use special technique which is called “AJAX” (Asynchronous Javascript And XML). The prominent point of AJAX is it supports asynchronous processing. Another issue user needs to keep in mind is any Web Applications require browser as its running environment. In this project, ASP.NET Web Form Application uses IIS, too.

Steps of working for ASP.NET Web Form Application are almost the same as (a) to (f) of a) except at (e) in Web Application, the submitted data will not show in Message Box. It will be in label format as pointed by the arrow cursor in Figure 7.

Automatic sending changed control data to Arduino every second can be done in the same way as in Classic Windows Form application. User just check the "Timer" CheckBox.

Table 2 shows sample of control data that are sent from client program (Windows or Web Application) to destination devices (LED's). Table 2 provides just 3 samples. In fact, because there are 16 bit Digital On/Off control signal choices, there can be 65,536 different results (2 raised to the 16th power combinations)

TABLE 2 SAMPLE RESULT OF DATA SENT FROM SIGNAL SOURCES TO DESTINATIONS ON CLIENT'S SCREEN

| **Sample #** | **Client Program (Windows or Web Application)** | | | **Web Server** | | **Microcontroller Parts** | **Electronic Parts** | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | ***SIPO Shift Registers*** | | ***Destination Devices (n=on, f=off)*** | |
| | ***Check Box #15-0 (c=checked, u=un checked)*** | ***Binary String*** | ***Hexa decimal String*** | ***Hexa decimal String (XML)*** | ***Binary String (XML)*** | ***Arduino*** | ***MSB*** | ***LSB*** | ***LED #15-8*** | ***LED #7-0*** |
| 1 | ccccuuuu cucucucc | 11110000 10101011 | F0AB | F0AB | 11110000 10101011 | F0AB | 11110000 | 10101011 | nnnn ffff | nfnf nfnn |
| 2 | uuucuucu cuucccuu | 00010010 10011100 | 129C | 129C | 00010010 10011100 | 129C | 00010010 | 10011100 | fffn ffnf | nffn nnff |
| 3 | cuuuuuuc uuuccuuu | 10000001 00011000 | 8118 | 8118 | 10000001 00011000 | 8118 | 10000001 | 00011000 | nfff fffn | fffn nfff |
| ===> Data Direction ===> | | | | | | | | | | |

2) Web Service Software

.NET Web Services part, from Figure 9, contain 4 Web Methods as follows.

a) Web Method for opening specified serial port

```
[WebMethod(Description = "Open Serial Port
COM#")]
public bool SerialPortOpen(string
strPortNumber)
```

This Web Method requires input as a string (serial port number) and returns output as a bool stating whether port opening succeeded or not.

b) Web Method for closing specified serial port

```
[WebMethod(Description = "Close Serial
Port")]
public bool SerialPortClose()
```

This Web Method returns output as a bool stating whether port closing succeeded or not.

c) Web Method for checking whether the specified port is available or not

```
[WebMethod(Description="Check whether port
is Avalable")]
public bool IsPortAvailable(string
strPortNumber)
```

This Web Method requires input as a string (serial port number) and returns output as a bool stating whether status of the specified port is available.

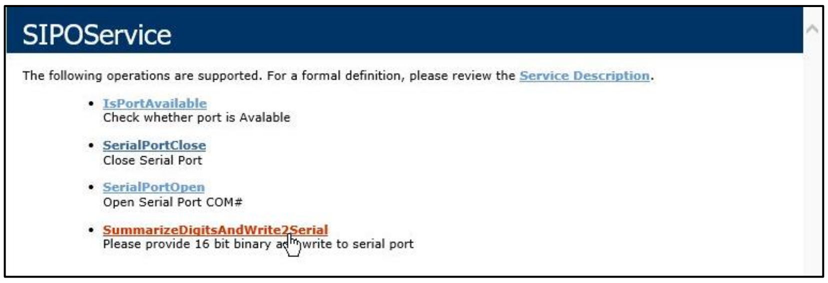
d) Web Method for getting data from series of CheckBox and send to serial port.

```
[WebMethod(Description="Please provide 16
bit binary and write to serial port")]
public string
SummarizeDigitsAndWrite2Serial(string
strBinStatus)
```

This Web Method needs binary string data as a parameter. The binary string data will be converted to hexadecimal and sent to serial port. The hexadecimal data that is written to serial port will be returned from the Web Method, too.

These Web Methods will be used by client applications. Testing page of the Web Methods in .NET Web Services that is provided by .NET framework is shown in Figure 9. In Figure 9, user wants to test "SummarizeDigitsAndWrite2Serial" Web Method that requires binary string as its parameter. After clicking, Figure 10 will show up. After filling binary string and click "Invoke" button, the result page will show up as Figure 11. Sample result of the Web Method in Web Services is shown in Figure 11. It will be in XML format. This XML data will be consumed by the client of Web Services

C# code of the SIPOService Web Services is shown in Figure 12. This Web Services codes are located in IIS Web Server.

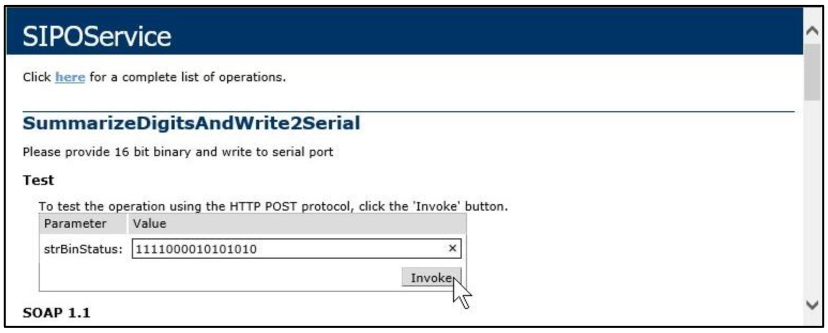


SIPOService

The following operations are supported. For a formal definition, please review the Service Description.

- IsPortAvailable
  Check whether port is Avalable
- SerialPortClose
  Close Serial Port
- SerialPortOpen
  Open Serial Port COM#
- SummarizeDigitsAndWrite2Serial
  Please provide 16 bit binary and write to serial port

Figure 9 List of Web Methods of SIPOService thouse are shown in test page



SIPOService

Click here for a complete list of operations.

**SummarizeDigitsAndWrite2Serial**

Please provide 16 bit binary and write to serial port

**Test**

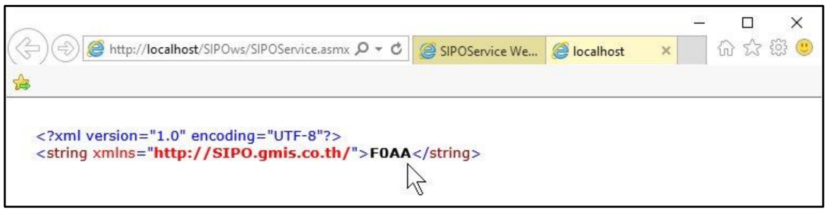
To test the operation using the HTTP POST protocol, click the 'Invoke' button.

| Parameter | Value |
|---|---|
| strBinStatus: | 1111000010101010 |

Invoke

**SOAP 1.1**

Figure 10 Web Method will provide the parameter input page if it needs parameter value.



http://localhost/SIPOws/SIPOService.asmx

```
<?xml version="1.0" encoding="UTF-8"?>
<string xmlns="http://SIPO.gmis.co.th/">F0AA</string>
```

Figure 11 Sample hexadecimal string data gotten from "SummarizeDigitsAndWrite2Serial" Web Method.

3) Arduino Software

Codes of Arduino are in Figure 13. Processes of Arduino Layer Software when getting Digital On/Off control data from Web Services via its serial port are as follows

(1) Getting Serial Data (in hexadecimal format) from Web Services
(2) Transform Serial Data to be Parallel Data
(3) Arrange High and Low Bytes to send Digital On/Off control signal to appropriated destination (LED's).

## V. CONCLUSION

The project can do as the objective which is sending multi-bit Digital On/Off control signals from multi-platform clients via standard Web Services through only one input port of Arduino to multiple destination devices. This project separates task to 3 parts, user inter face part (client software), business logic part (Web Services) and microcontroller with device part (Arduino). Sample multi-platform clients in this project are Classic Windows Form and ASP.NET Web Form applications. Steps of working are as follows. Any platform clients, that support Web Services, send multi-bit Digital On/Off control data to Web Services. Web Services sends that serial data to Arduino. (The reason that data should be in serial format is it will consume just one output port at Arduino board.) Arduino with the cooperation of SIPO Shift Registers (CD4094BE) transforms serial data to parallel data and distributes to appropriated digital devices. By using SIPO Shift Registers, customers can cascade more modules which can extend amount of control data bit and destination devices as need.

## VI. FUTURE WORK

There are many issues that can do further from this project. First, creating clients in other platforms, such as, java, android and iOS. Second, wrapping data communicated in the whole project with standard industrial bus, such as Modbus TCP.

## REFERENCES

[1] H. Haas and A. Brown, "Web Services Glossary. W3C," *W3C Working Group Note*, 2004. [Online]. Available: http://www.w3.org/TR/ws-

gloss/. [Accessed: 09-Sep-2015].

[2] "Getting Started with Arduino," 2016. [Online]. Available: https://www.arduino.cc/en/Guide/HomePage. [Accessed: 11-Apr-2016].

[3] "Multiplexer and Demultiplexer," 2013. [Online]. Available: http://www.electronicshub.org/multiplexer-and-demultiplexer/. [Accessed: 08-Sep-2015].

[4] "Shift Registers," 2014. [Online]. Available: http://www.ee.usyd.edu.au/tutorials/digital_tutorial/part2/register01.html. [Accessed: 08-Sep-2015].

[5] P. Lau, "Serial In-Parallel Out Shift Register," *School of Electrical & Information Engineering, University of Sydney*, 2010. [Online]. Available: https://www.ee.usyd.edu.au/tutorials/digital_tutorial/part2/register03.html. [Accessed: 11-Apr-2016].

## APPENDIX

Essential codes of the project come from parts of Web Services and Arduino are listed in Figure 12 and 13 respectively.

```
using System;
using System.Collections.Generic;
using System.IO.Ports;
using System.Web;
using System.Web.Services;
namespace LearnShiftOutClientWS01Prj  {
   [WebService(Namespace = "http://SIPO.gmis.co.th/")]
   [WebServiceBinding(ConformsTo =
WsiProfiles.BasicProfile1_1)]
   [System.ComponentModel.ToolboxItem(false)]
   [System.Web.Script.Services.ScriptService]
   public class SIPOService :
System.Web.Services.WebService  {
      static SerialPort srpMain = null;
      static bool blnIsSerialPortOpen = false;
      public static SerialPort SrpMain  {
         get { return srpMain; }
         set { srpMain = value; }
      }
      public static string StrBinStatus { get; set; }
      public static string StrHexStatus { get; set; }
      public static string StrCOMPort { get; set; }
      public SIPOService()  {}
      public string ConvertBinStringToHexString(string
strBinString)  {
         int i = Convert.ToInt32(strBinString, 2);
         string strHex = i.ToString("X4");
         return strHex;
      }
      [WebMethod(Description="Please provide 16 bit binary and
write to serial port")]
      public string SummarizeDigitsAndWrite2Serial(string
strBinStatus)  {
         string strHexStatus = "";
         strHexStatus =
ConvertBinStringToHexString(strBinStatus);
         if (!srpMain.IsOpen) { srpMain.Open(); }
         srpMain.WriteLine(strHexStatus);
         srpMain.Close();
         StrBinStatus = strBinStatus;
         StrHexStatus = strHexStatus;
         return strHexStatus;
      }
      [WebMethod(Description="Check whether port is
Avalable")]
      public bool IsPortAvailable(string strPortNumber)  {
         if (SrpMain.PortName == "COM" + strPortNumber)  {
            return blnIsSerialPortOpen;
         } else {
            return false;
         }
      }
      [WebMethod(Description = "Open Serial Port COM#")]
      public bool SerialPortOpen(string strPortNumber)  {
         try {
            srpMain = new SerialPort();
            SrpMain = srpMain;
            SrpMain.PortName = "COM" + strPortNumber;
            SrpMain.BaudRate = 9600;
            SrpMain.Parity = Parity.None;
            SrpMain.DataBits = 8;
            SrpMain.StopBits = StopBits.One;
            SrpMain.DtrEnable = true;
            SrpMain.Open();
            StrCOMPort = SrpMain.PortName;
            if (SrpMain.IsOpen == true)
               blnIsSerialPortOpen = true;
            return blnIsSerialPortOpen;
         }  catch  {
            srpMain.Dispose();
            srpMain.Close();
            blnIsSerialPortOpen = false;
            return blnIsSerialPortOpen;
         }
      }
      [WebMethod(Description = "Close Serial Port")]
      public bool SerialPortClose()  {
         bool blnIsSerialPortClose = false;
         SrpMain.Close();
         if (SrpMain.IsOpen != true)  {
            blnIsSerialPortClose = true;
         }
         return blnIsSerialPortClose;
      }
   }
}
```

Figure 12  Web Services Code (SIPOService) (LearnShiftOutClientWS01.asmx.cs)

```
int intStobePin, intClockPin, intDataPin;
int intToggle = -1;
int intInputMaxDigit = 16; // 16 bit
int UIntDataOut = 0;
char chrBuffer[4];
void ShiftOut(int intDataPin, int inClockPin, u16 bytDataOut, int
intMaxDigit) {
        int intPinState = 0;
        for (int i = intMaxDigit; i >= 0; i--) {
                digitalWrite(intClockPin, 0);
                if (bytDataOut & (1 << i)) {
                        intPinState = 1;
                }
                else {
                        intPinState = 0;
                }
                digitalWrite(intDataPin, intPinState);
                digitalWrite(intClockPin, 1);
                digitalWrite(intDataPin, 0);
        }
        digitalWrite(intClockPin, 0);
}
void setup() {
        intStobePin = 10;
        intClockPin = 11;
        intDataPin = 12;
        pinMode(intStobePin, OUTPUT);
        pinMode(intClockPin, OUTPUT);
        pinMode(intDataPin, OUTPUT);
}
void loop() {
        u16 UIntDataGet = 0;
        int intT = Serial.available();
        if (intT > 0) {
                Serial.readBytesUntil('\n', chrBuffer,
Serial.available());
                UIntDataGet = strtol(chrBuffer, NULL, 16);
                Serial.print(UIntDataGet, BIN);
                Serial.print(" - ");
                Serial.println(UIntDataGet, HEX);
                digitalWrite(intStobePin, LOW);
                ShiftOut(intDataPin, intClockPin,
UIntDataGet, intInputMaxDigit);
                digitalWrite(intStobePin, HIGH);
                delay(500);
        }
}
```

Figure 13 Arduino Code (LearnShiftOut03Prj.ino)